Assesment-01 Junit

Kshitij kumar-12214966

LoanService.java

```
package org.example;

public class LoanService {

  public boolean isEligible(int age, double salary) {

  if (age < 21 || age > 60) {
  return false;
  }

  if (salary < 25000) {
  return false;
  }

  return true;
  }

  public double calculateEMI(double loanAmount, int tenureYears)
{

  if (loanAmount <= 0) {
  throw new IllegalArgumentException("Invalid loan amount");
  }

  if (tenureYears <= 0) {
  throw new IllegalArgumentException("Invalid tenure");
  }

  return loanAmount / (tenureYears * 12);
  }

  public String getLoanCategory(int creditScore) {

  if (creditScore >= 750) {
  return "Premium";
  }
  else if (creditScore >= 600) {
  return "Standard";
  }
```

```
  else {
  return "High Risk";
  }
  }
}
```

LoanServiceTest.java

```
package org.example;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class LoanServiceTest {
  private LoanService loanService;

  @BeforeEach
  public void setup(){
  loanService = new LoanService();
  }

  @Test
 @DisplayName("Test isEligible")
  public void testIsEligible(){
  assertTrue(loanService.isEligible(22, 30000));
  }

  @Test
 @DisplayName("Test Age")
  public void testInvalidAge(){
  assertAll(
  () -> assertFalse(loanService.isEligible(17, 30000)),
  () -> assertFalse(loanService.isEligible(61, 30000))
  );
  }

  @Test
 @DisplayName("Test salary")
  public void testSalary() {
  assertAll(
  () -> assertFalse(loanService.isEligible(21, 20000)),
```

```
() -> assertTrue(loanService.isEligible(21, 35000))
);
}

@Test
@DisplayName("Test calculateEmi loanAMount")
public void testCalculateEmi(){
assertThrows(IllegalArgumentException.class, () -> {
loanService.calculateEMI(0, 10);
});
}

@Test
@DisplayName("Test calculateEmi teanureYears")
public void testEMIYears() {
assertThrows(IllegalArgumentException.class, () -> {
loanService.calculateEMI(2, 0);
});
}

@Test
@DisplayName("Test EMi calculation")
public void testEmiCalculation(){
double emi = loanService.calculateEMI(120000, 1);
assertEquals(10000.0, emi);
}

@Test
@DisplayName("Test Premium Category")
public void testPremiumCategory() {
assertEquals("Premium", loanService.getLoanCategory(800));
}

@Test
@DisplayName("Test Standard Category")
public void testStandardCategory() {
assertEquals("Standard", loanService.getLoanCategory(700));
}

@Test
@DisplayName("Test High Risk Category")
public void testHighRiskCategory() {
assertEquals("High Risk", loanService.getLoanCategory(500));
}

@Test
```

```java
 @DisplayName("Boundary Values for creditScore")
  public void testBoundaryValues(){
  assertAll(
  () -> assertEquals("Premium",
loanService.getLoanCategory(750)),
  () -> assertEquals("Standard",
loanService.getLoanCategory(600)),
  () -> assertEquals("High Risk",
loanService.getLoanCategory(599))
  );
  }

  @Test
 @DisplayName("Test notnull")
  public void testNotNull() {
  assertNotNull(loanService);
  }


}
```